NAME OF THE PROGRAM : 1

NO.

DATE 28/08/2025

1. Write a program to implement tic tac toe game using python.

```
def print_board(board):
    for row in board:
        print("|".join(row))
        print("-"*5)
def check_winner(board, player):
    #chek rows
    for row in board:
        if all(cell == player for cell in row):
            return True
    #check columns
    for col in range(3):
        if all(board[row][col] == player for row
in range(3)):
            return True
    # check diagonals
    if all(board[i][i] == player for i in range(3)) or
    all(board[i][2-i] == player for i in range(3)):
        return True
    return False
```


PSN COLLEGE OF ENGINEERING | PSN COLLEGE OF ENGINEERING | PSN COLLEGE OF ENGINEERING | PSN COLLEGE OF ENGINEERING | PSN COLLEGE OF ENGINEERING

Output:

Welcome to Tic-Tac-Toe!

Player x's turn

Enter the row (0, 1 or 2): 0
Enter the column (0, 1 or 2): 0

Player o's turn

Enter the row (0, 1 or 2): 1
Enter the column (0, 1 or 2): 1

Player x's turn

Enter the row (0, 1 or 2): 2
Enter the column (0, 1 or 2): 2

Player o's turn

Enter the row (0, 1 or 2): 0
Enter the column (0, 1 or 2): 1

Player x's turn

Enter the row (0, 1 or 2): 2
Enter the column (0, 1 or 2): 1

Player o's turn

Enter the row (0, 1 or 2): 1
Enter the column (0, 1 or 2): 2

Player x's turn

Enter the row (0, 1 or 2): 0
Enter the column (0, 1 or 2): 2

Player o's turn

Enter the row (0, 1 or 2): 1
Enter the column (0, 1 or 2): 0

Player x's turn

Enter the row (0, 1 or 2): 1
Enter the column (0, 1 or 2): 0

NAME OF THE PROGRAM	NO
	DATE

```
def main():
    board = [" " for _ in range(3)] for _ in range(3)]
    players = ["x", "o"]
    turn = 0
    print("Welcome to Tic-Tac-Toe")
    print_board(board)
    while True:
        player = players[turn%2]
        print(f"player {player}'s turn")
        row = int(input("Enter row (0,1 or 2):"))
        col = int(input("Enter column (0,1 or 2):"))
        if board[row][col] == " ":
            board[row][col] = player
            print_board(board)
            if check_winner(board, player):
                print(f"player {player} wins!")
                break
            elif is_board_full(board):
                print("It's a draw.")
                break
            turn+=1
        else:
            print("That cell is already occupied.
                  Try again")
if __name__ == "__main__":
    main()
```

```
initial state = [0,0]
capacities = (4,3)
goal state = [2,y]
enter the rule : 2
0 3
enter the rule : 8
3 0
enter the rule : 2
3 3
enter the rule : 6
4 2
enter the rule : 3
0 2
enter the rule : 9
2 0
```

NAME OF THE PROGRAM 2 | NO

DATE 4/08/2015

2] Write a program to implement water-jug problem using Python.

```
x=0
y=0
m=4
n=3
print("initial state = (0, 0)")
print("capacities = (4,3)")
print("goal state = (2,y)")
while (x!=2):
    n = int(input("enter the rule: "))
    if (n==1):
        x = m
    elif (n==2):
        y = n
    elif (n==3):
        x=0
    elif (n==4):
        y = 0
    elif (n==5):
        t = n-y
        y = n
        y -= t
```

PAGE NO 3

NAME OF THE PROGRAM | NO. | DATE

```
elif (n==6):
    t = m-x
    x = m
    y = t
elif (n==7):
    y = x
    x = 0
elif (n==8):
    x = y
    y = 0
else:
    print ("invalid rule")
print (x,y)
```

NAME OF THE PROGRAM 3	NO.
	DATE

3) write a program to implement 8puzzle problem using python

```
import numpy as np
import pandas as pd
import os
def bfs(src, target):
    queue = []
    queue.append(src)
    exp = []
    while len(queue)>0:
        source = queue.pop(0)
        exp.append(source)
        print(source)
        if source == target:
            print("success")
            return
    poss_moves_to_do = []
    poss_moves_to_do = possible_moves
                        (source, exp)

    for move in poss_moves_to_do:
        if move not in exp and move not
                                in queue:
            queue.append(move)
```

NAME OF THE PROGRAM | NO. | DATE

```
def possible_moves(state, visited_states):
    b = state.index(0)
    d = []
    if b not in [0,1,2]:
        d.append('u')
    if b not in [6,7,8]:
        d.append('d')
    if b not in [0,3,6]:
        d.append('l')
    if b not in [2,5,8]:
        d.append('r')
    poss_moves_it_can = []
    for i in d:
        poss_moves_it_can.append(gen(state,i,b))
    return [move_it_can for move_it_can in
            poss_moves_it_can if move_it
            _can not in visited_states]
def gen(state, m, b):
    temp = state.copy()
    if m == 'd':
        temp[b+3], temp[b] = temp[b],
    if m == 'u':                    temp[b+3]
        temp[b-3], temp[b] = temp[b], temp[b-3]
    if m == 'l':
        temp[b-1], temp[b] = temp[b], temp[b-1]
```

PAGE NO.

LEGE OF ENGINEERING • PDA COLLEGE OF ENGINEERING • PDA COLLEGE OF ENGINEERING • PDA COLLEGE OF ENGINEERING • PDA COLLEGE OF ENGINEERING • PDA COLLEGE OF ENGINEERING

NAME OF THE PROGRAM | NO. | DATE

```
if m=='r':
    temp[6+1], temp[6] = temp[6], temp[6+1]
return temp
src = [1,2,3,4,5,6,0,7,8]
target = [1,2,3,4,5,6,7,8,0]
bfs(src, target)
```

output :-

[1, 2, 3, 4, 5, 6, 0, 7, 8]

[1, 2, 3, 0, 5, 6, 4, 7, 8]

[1, 2, 3, 4, 5, 6, 7, 0, 8]

[0, 2, 3, 1, 5, 6, 4, 7, 8]

[1, 2, 3, 5, 0, 6, 4, 7, 8]

[1, 2, 3, 4, 0, 6, 7, 5, 8]

[1, 2, 3, 4, 5, 6, 7, 8, 0]

success